Volume 5, Issue 5, May 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Implementation of Efficient Extraction of Deep Web by Applying Structured Queries and Maintaining Repository Freshness

Yogesh Kakde, Manoj Kumar Rawat, Jitendra Dangra
LNCT Indore, Madhya Pradesh,
India

Abstract— Information on web is growing at extremely rapid rate. To retrieve this huge amount of information a lot of search engines are used. Information on the web which cannot be indexed by traditional search engines is called deep web while, the amount of information in the web which can be easily indexed by traditional search engine is called Surface Web. Retrieval of data from the contents of deep web databases requires some additional process which includes filling out deep web databases access points, i.e., filling forms in the web and construction of structured queries. Traditional search engines are not capable to fill out the forms and submit the queries automatically, and consequently, they are not capable to access or index the information from deep web straight.
Researches intend that the majority of the rich and high excellence of information is nearby in the deep. A number of techniques are proposed to solve this problem. We have proposed a novel technique which is capable to build a suitable query which is designed especially for deep web data and an additional module which works after building query and extracts data from deep web source.
When data from deep web is retrieved we manage the copy of that data in Data Repository with having a number of attributes. We also manage this data repository refreshed by implementing another algorithm. This repository works as a local cache in our project and makes data retrieval process more efficient.

Kew words: Deep web, crawling, Query Building, Data Repository, Data Extraction

I. INTRODUCTION

The Deep Web also called the Deep net/Invisible net [9] or Hidden Web is the content of World Wide Web that does not belong from the upper layer Surface Web, which is accessed and indexed by the search engines. It should not be mystified with the dark Internet; refers to web pages that cannot be indexed by a typical search engine or Web pages that a search engine calculatedly does not index in its DB, interprets the data which is invisible to the end-user. Simple search engines can-not retrieve contents from the deep web.

In order to solve the problem of retrieving the information from the private domains of the user and displaying the desired result to the user, we propose a framework. In this project there are various terminologies such as crawler, deep web etc. So before proceeding towards the project explanation, we would like to give you brief idea about deep web and crawling process. The deep or invisible or hidden web is the hidden part of the Web, which is very difficult to crawl by conventional search engines. Research [11] tells that the majority of the rich and high quality of information is present in the deep Web and is originated from web databases. As described in the white paper in Bright Planet [23], the amount of web databases server sites has reached to the range from 50,000 to 1, 00,000 and is increasing day after day with rapid very fast rate of growth. The information stored in the web is estimated about 8000 terabytes and the amount of the information found in the deep Web is about 550 times of the Surface Web [23]. Information in hidden web databases are usually differentiated whose representation is un-structured, which is vast and may or may not have subject-oriented content. The extraction, retrieving, mining and integration of the relevant Information from the web databases are difficult for various applications. Web crawlers are those programs that travel the web on behalf of the search engine, and follow proper links to reach various web pages to download them. The process of crawling begins with a set of seed URLs, the crawler extracts URLs showing in the retrieved and accessed pages, and resultant filtered pages are stored in database.

Data repository is a large collection of data object which stores and manages having various essential attributes. All the information that are been crawled by crawler and discover applicable are retrieved and stored in the data repository. Our proposed repository in the project acts as the local cache for this information retrieval system.

II. RELATED WORK

Md. Abu Kausar in at al [1] proposed approach uses Java Aglets for crawling of the web pages. The major gain of web crawler which is based on mobile agents system are that the explication part of the crawling process is actually done at the local residence of data rather than inside the remote server. This paper explains a web page modification detection technique which based on web page index using mobile agents.

Meenakshi Sharma and Supriya in [2] have explained the query building algorithm. They explained that there are lots of this information is contained in the databases and is not been indexed by search engines. As the Deep web data is

immersed deep inside the various databases and hidden behind the forms of search query, this information can only be retrieved only by interacting with these forms. This paper proposed a new approach to interact with search forms and extract deep web data by employing a strategy using query building technique.

Nripendra Narayan Das and Ela Kumar in [3] have described the algorithm for extraction of data from deep web. They described that the general manual method is not appropriate for huge number of pages. It is a challenging work to access and retrieve appropriate and useful information from web pages. Currently, many web mining systems called web wrappers, web crawler have been designed. In this paper, some already existing techniques are analyzed and then our new proposed work on web information extraction is explained. This is a fact that many search engines are not capable to extract the data from deep web as they lack by some logical issues for extraction of data. In this paper an algorithm has been defined and implemented which will extract the data in efficient manner as well as display as a single page output.

Rahul Choudhari, R.D. Choudhari and Ajay Choudhari in [4] have shown the algorithm to increase the efficiency of search engines. They explained the scheduling problem and given a perfect solution for crawlers, with the objective in focus is to optimize the resources and improving quality of the indexes for web pages. Apart from this they have divided the web content resources in two portions: 1) Active 2) Inactive. For the content which is inactive content providers they used crawling agents who continuously crawls the resource of content providers and collect the pattern of updating of the content providers. They have also proposed a scheduling method which takes advantage on the information taken by the web agents.

Umara Noor, Zahid Rashi, Azhar Rauf in [7] have explained that A large part of Deep web comprises of online structured and domain specific databases. These databases are accessed using web query interfaces which is normally a web page. The information contained and saved in these databases is usually related to a particular domain. This highly relevant and useful information is the most suitable information for satisfying the needs of the users and for the implementation of large scale deep web integration

III. PROPOSED METHODOLOGY

The main issue in retrieving information from deep web is how to extract all the relevant information from the web in more efficient manner.

Here we have proposed system architecture.

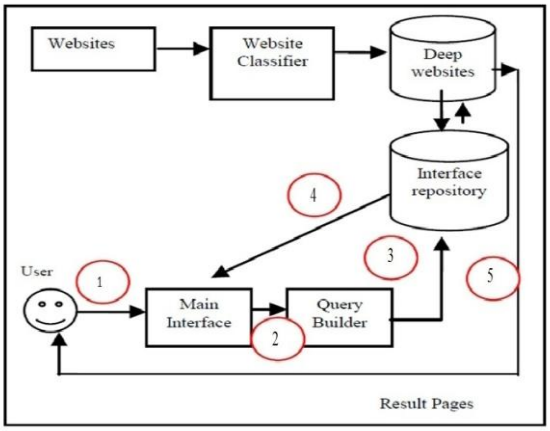

Figure 1: Architecture of Deep Web Data Mining System

Our proposed solution is based on three different modules these are-

A) Query Builder Module.
B) Data Extraction Module.
C) Refresh Repository Module.

A) Query Builder Module: Here we are implementing an algorithm for query building. We need to build a query which must have all the attributes that are to be retrieved from the web.

B) Data Extraction Module: We are again using an algorithm for the process of extraction of data from deep web.

C) Refresh Repository Module: After each extraction operation of data we perform a refresh repository operation so as to maintain the repository fresh a remove stale web pages.

Here we are giving complete explanation of all three modules discussed above. We will discuss about the algorithms that we have implemented in our project.

A) Query Builder Module: [2]

As the Deep web information is buried deep inside the databases and hidden behind the search query forms, this information can only be accessed by interacting with these forms.

Our propose technique converts free-text queries into structured queries that are suitable for a quantity of deep web site which has a form with numerous input fields. The interface appears like an easy search engine so that the user can unreservedly select fields for extraction to search. Relative user study will then perform task to review how users terminate a predefined set of search tasks with a standard system and the recently build technique.

Query Builder: User fills the Base interface form with his/her query, Query Builder processes, forms the query string URL and then submits the form to find the result page. Algorithm for query building is shown below.

B) Data Extraction Module:

It is the fact that most [3] of the search engines are not in a position to extract the data from deep web. In this paper an algorithm has been designed and practically implemented which will not only extract the data in efficient manner but will also display as a single page output.

The structured information usually comes from databases, which provide rigid or well defined formats of information, therefore, it is easy to extract through some query language such as Structured Query Language (SQL).

The other type is the semi structured information, which falls between free text and structured information. A good example of semi-structured information is Web pages.

Currently, the targeted web resources can be obtained easily by typing and then inputting some keywords with a web search engine. But the biggest problem is that the system may or may not provide relevant and useful data rich pages and it is very difficult for the computer to automatically extract the information contained. The reason behind this is that web pages are designed for human browsing and interpretation, rather than machine interpretation.

C) To keep the freshness **[4]** of the result by the search engine, crawling of the web page should be fundamentally linked with the frequency updates of the web pages. When a pattern for the page is being achieved it is given to parallel crawlers of the search engine for the crawl according to the schedule. This will keep the freshness of repository stable over the time and incrementally updated. Below there is crawling algorithm for the agents

Our task initiates with, the Input Queue which contains a list of seed URLs as input from the URL Database and the Extractors repeatedly follow and execute the steps declared. Access a URL, eliminate that URL from the queue, download the equivalent information contained in that URL, and extract any links contained in it. [4] Our project includes three different algorithms that are been include in our project just to implement efficient extraction of deep web. [2]

All the algorithms are explained below:

Algorithm query_build (URL list)

- Pick one by one URL from URL list
- Extract the values of all attributes.
- Append URLs with the values of these attributes. // URL/A1/A2/A3/……
- Submit URL.
- Get the result pages.

The algorithm for extraction of data:

- When query is submitted in search box, it matches in the database where data is stored of different domain with key attribute.
- Internally access the database and searches data in the XML code.
- If attribute matches with the attribute available in database it stores it in repository.
- Display the result in a particular format.
- The process is applied for all the databases and finally result is displayed in single web page.

Algorithm for maintaining freshness of repository:

```
Initialize:
UrlsDone = φ
UrlsToDo = { "intial set of urls for agent crawling
from INACTIVE DIRECTORY"}
Repeat:
Url = UrlsToDo.getNext()
Ip = DNSlookup(Url.getHostName())
If Url ∈ACTIVE DIRECTORY
Then break; Else Html= DownloadPage(ip,Url.getPath())
If Html ∉Repository Save (url, html)
Else Udate(Url, html)
Repository = Old Urls + new Urls – common Urls Tmp = saved pages( html.getTime)
Till tmp = C (constant) /*C represents a particular pattern*/
Send tmp to scheduler UrlsDone.insert(url)
newUrls = parsefor links(html) for each newurl if not UrlsDone.contains(new url)
then urlsToDo.insert(new url)
```


IV. IMPLEMENTATION

We have use three different data source. These are guru.com, odesk.com, freelancer.com. We have applied our proposed approach and work live by accessing direct API link and extracting XML data from these three website.

V. RESULT ANALYSIS

Evolution of result requires lots of analysis of various object like functions & methods, reference & implemented classes, function behavior, CPU utilization, memory consumed etc. We have first of all generated some report of function call stack. Since we have designed the solution of our project in the form of combination of functions, which contains the implementation of the algorithm studied.

Sampling: The sampling profiling method which is a function to generate reports is the Visual Studio Profiling Tools which interrupts the computer processor at set intervals and then collects the function call stack.

Summary View - Profiler Sampling Data:

This summary view tool gives information of the functions that are most performance-expensive methods. By using this summary view we can also find the functions that were performing the most individual work. By using this tool we can also find the CPU usage value by the function that is being executed when any computational operation is performed.

Here we have drawn a graph of CPU utilization while we were extraction data from guru.com (guru.com is simply an example. We might use any other web source too).

Timeline Graph: This part of summary view gives the percents (%) of CPU utilization of the profiled application over the time when the profiling is done. We can also select a time span to filter the view of the timeline graph.

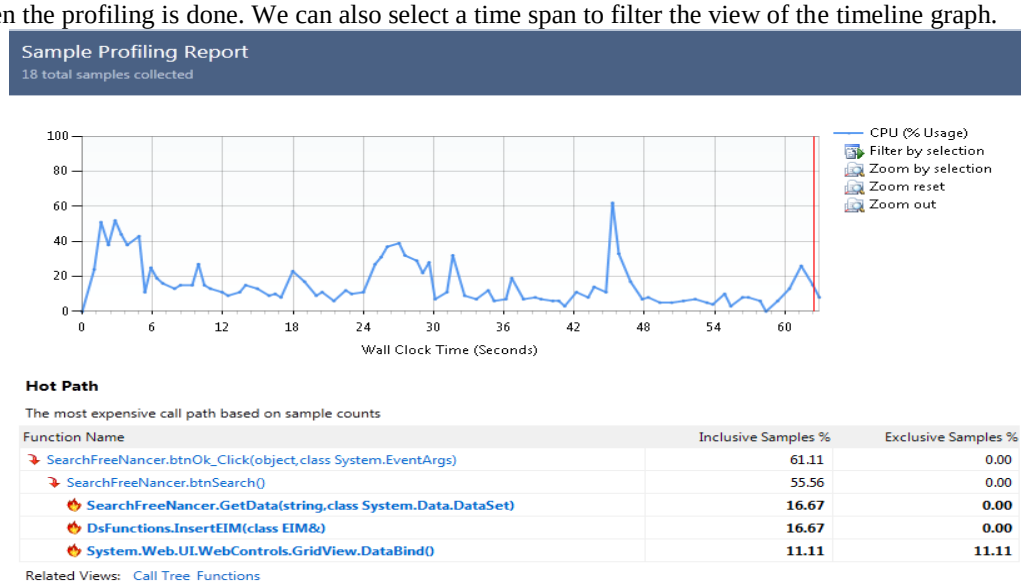

Fig.2: Analysis of Summary View for Data Extraction from Guru.com

Function Detail View:

Apart from this we have also analyzed the behavior of btnSearch() function for guru.com. We have chosen this function because this is the function which contains the implementation of the data extraction algorithm [3].

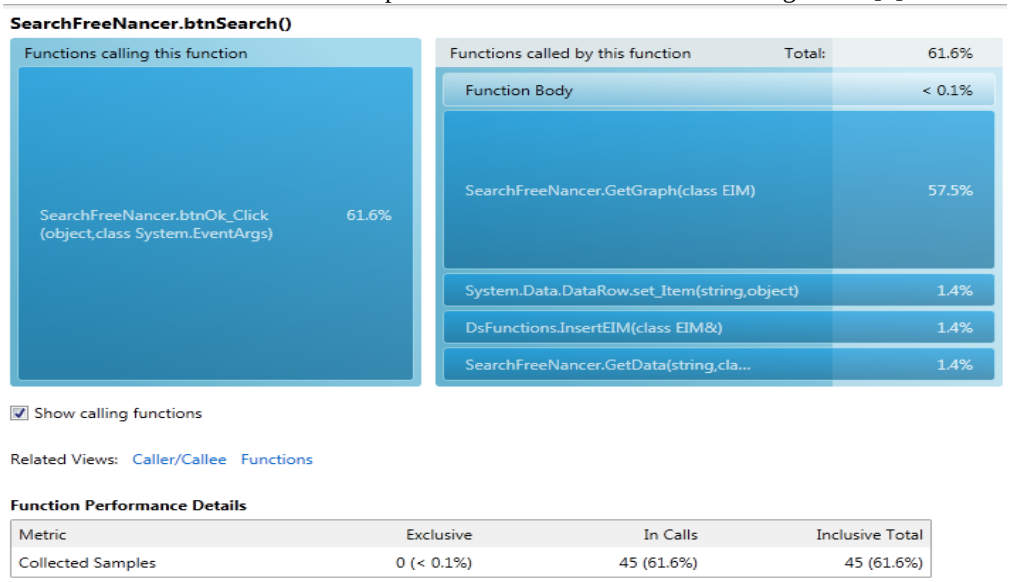

Fig.3: Function Details View

Memory Consumption Report:

We have designed report which shows the memory consumed by some special functions in our implementation. This time we have taken two important functions to find their memory requirement. We have designed function detail view of both the functions. These are as below-

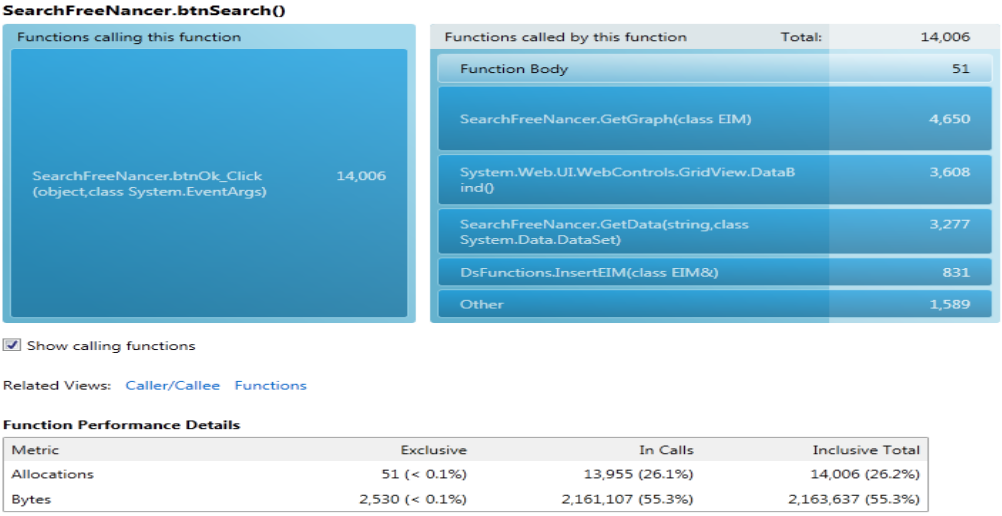

Fig.4: Memory Profiling Report for function btnsearch()

VI. COMPARISON

Here in this we are showing a graph which compares the data accessed for deep web data retrieval and data actually extracted from remote server. Since we have merged different algorithms from different research studies and have implemented, so we confidently say that our proposed implementation does not leave a single data from database which is being demanded by user.

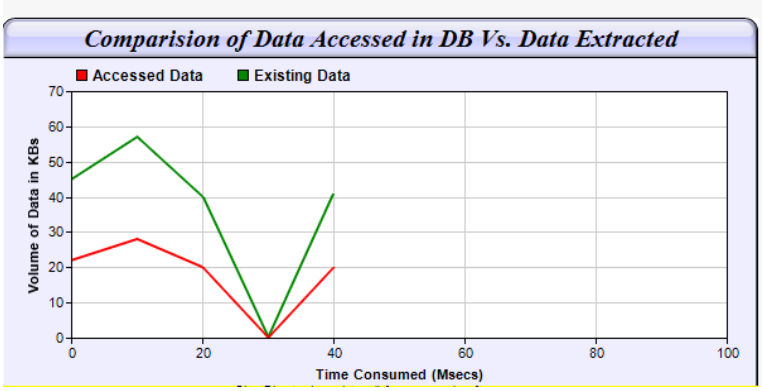

Fig.5: Graph for Guru.com

VII. CONCLUSION AND FUTURE WORK

We have analyzed the result in form of CPU Utilization and memory allocated for the functions which are implementation of algorithms. So we found that

Average CPU utilization for the operation of retrieving data from guru.com (Query Design and Data Extraction) is about 25%.

Memory Consumption by the btnsearch() function for extraction of data is 26% of all inclusive samples.

Memory Consumption by the btncreatequery() function for extraction of data is 14% of all inclusive samples.

We have implemented the combination of algorithm in different logical functions. In future we can merge the codes of query building and data extraction so as to make it more efficient in terms of memory consumption and CPU utilization.

ACKNOWLEDGMENTS

I am thankful to the authors of [2], [3] and [4]. In this research we have referred three different algorithms from these papers and combined them with little modifications. Thanks to Dr. Manoj Kumar Rawat and Prof. Jitendra Dangra, Computer Science & Engg Dept, LNCT Indore who have contributed towards development of the project.

REFERENCE

[1] Md. Abu Kausar , Md. Nasar , Maintaining The Repository of Search Engine Freshness Using Mobile Crawler,. International Conference on Microelectronics, Communication and Renewable Energy (ICMiCR-2013).

[2] Supriya, Minakshi Sharma, Deep Web Data Mining, International Journal of IT, Engineering and Applied Sciences Research (IJIEASR) ISSN: 2319-4413(March 2013)

[3] Nripendra Narayan Das, Ela Kumar, AUTOMATIC EXTRACTION OF DATA FROM DEEP WEB PAGE, International Journal of Computer & Mathematical Science ISSN: 2347-8527,Apr 2014

[4] Choudhari, R.,, Increasing Search Engine Efficiency Using Cooperative Web,, Computer Science and Software Engineering, 2008 International Conference on (Volume:4) on IEEE.

[5] Yogesh kakde, Dr. Manoj Rawat, "Accumulative Search Engine Effectiveness Using Supportive Web" IJSHRE, 2014, 2347-4890.

[6] Yogesh kakde, Dr. Manoj Rawat, Mr. Jitendra Dangra, Novel Approach for Data Source Integration System Update Strategy in Hidden Web, Engineering Universe for Scientific Research and Management, Vol. 7 Issue 2 February 2015.

[7] Umara Noor, Zahid Rashid, Azhar Rauf, A Survey of Automatic Deep Web Classification technique, International Journal of Computer Applications (0975 – 8887) Volume 19– No.6, April 2011

[8] Dilip Kumar Sharma, A. K. Sharma, Analysis of Techniques for Detection of deep web search interface , CSI COMMUNICATIONS | DECEMBER 2010

[9] Md. Abu Kausar, V. S. Dhaka, Sanjeev Kumar Singh, Web Crawler A Review, International Journal of Computer Applications (0975 – 8887) Volume 63– No.2, February 2013

[10] T Manoj D. Swami, Gopal Sonune, Dr. B.B. Meshram, Understanding the Technique of Data Extractionfrom Deep Web, International Journal of Computer Science and Information Technologies, Vol. 4 (3) , 2013, 533-537

[11] Jufeng Yang Guangshun Shi Yan Zheng Qingren Wang, Data Extraction from Deep Web Pages, International Conference on Computational Intelligence and Security, IEEE 2007

[12] Carlos Castillo, Mauricio Marin, Andrea Rodriguez, Ricardo Baeza-Yates, Scheduling Algorithms for Web Crawling,

[13] Alsayed Algergawy, Gunter Saake, Combining Multiple Features for Web data source clustering

[14] Divya Dalal, Deep Web Query Extraction Algorithm for Information Retrieval System, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (5) , 2014, 6867-6870

[15] Ritu Khare Yuan An Il-Yeol Song," Understanding Deep Web Search Interfaces: A Survey" SIGMOD Record, March 2010 (Vol. 39, No. 1)

[16] Soniya Agrawal, Dharmesh dubey, Efficient Approach for Knowledge Management Using Deep Web Information Retrieval System, IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661, p-ISSN: 2278-8727Volume 12, Issue 2 (May. - Jun. 2013), PP 93-100

[17] Kapil Sahu, Deepali Kelkar, Enhancing Information Retrieval by Integration Invisible web Data Source. International Journal of Engineering and Innovative Technology (IJEIT),Volume 2, Issue 4, October 2012

[18] Babita Ahuja, Anuradha, Ashish Ahuja, Hidden Web Data Extraction Tools, International Journal of Computer Applications (0975 – 8887), Volume 82 – No 15, November 2013

[19] Priyanka Jain, Megha Bansal, Efficient Crawling the Deep Web, International Journal of Advanced Research in Computer Science and Software Engineering ISSN: 2277 128X, May 2014

[20] Trupti V. Udapure, Ravindra D. Kale, Rajesh C. Dharmik, Study of Web Crawler and its Different Types, IOSR Journal of Computer Engineering (IOSR-JCE)

[21] Soniya Agrawal, Dharmesh Dubey, Novel Query Planning Approach for Deep Web Information Retrieval System, International Journal of Engineering Research & Technology (IJERT) Vol. 2 Issue 11, November – 2013

[22] Umara Noor, Ali Daud, Ayesha Manzoor, Latent Dirichlet Allocation based Semantics Clustering of Heterogenous Deep Web Sources, 2013 5th International Conference on Intelligent Networking and Collaborative Systems, IEEE

[23] MICHAEL K. BERGMAN, The Deep Web: Surfacing Hidden Value, WHITE PAPER, BrightPlanet - Deep Content, September 24, 2001

[24] CHARACTERISTICS OFTHE INVISIBLE WEB, Going Beyond Google: The Invisible Web in Learning and Teaching,

[25] Parvatam Sriram Rohit, R. Krishnaveni, Deep Malicious Website Detection, International Journal of Computer Science and Mobile Computing, ISSN 2320–088X Vol. 2, Issue. 4, April 2013, pg.517 – 522

[26] Hui Li, Mei Guo, Liang Cai ," An Incremental Update Strategy in Deep Web" 2010 Sixth International Conference on Natural Computation (ICNC 2010) 978-1-4244-5961-2/10/$26.00 ©2010 IEEEBergman, M. K., "The Deep Web: Surfacing Hidden Value," White paper, 2001.

[27] Jesús S. Aguilar-Ruiz, Raúl Giráldez, and José C. Riquelme , Natural Encoding for Evolutionary Supervised Learning, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, VOL. 11, NO. 4, AUGUST 2007.

[28] Jaideep Srivastava, Prasanna Desikan, Vipin Kumar, Web Mining— Concepts, Applications, and Research Directions.

[29] Bin He, Tao Tao, Kevin ChenChuan Chang, Organizing Structured Web Sources by Query Schemas: A Clustering Approach, ACM, 2004

[30] Yao-Wen Huang, Shih-Kun Huang, Tsung-Po Lin, Chung-Hung Tsai, Web Application Security Assessment by Fault Injection and Behavior Monitoring, ACM, 2003

